Improved Parallel Prefix Algorithm on OTIS-Mesh of Trees

Ashish Gupta
Department of Computer Science and Engineering
Indian School of Mines, Dhanbad, Jharkhand, 826004, India
Email: ashish.parj@gmail.com

***Abstract*—A parallel algorithm for prefix computation reported recently on interconnection network called OTIS-Mesh Of Trees[4]. Using n^4 processors, algorithm shown to run in 13log n + O(1) electronic moves and 2 optical moves for n^4 data points. In this paper we present new and improved parallel algorithm for prefix on OTIS-Mesh of Trees. The algorithm requires 10log n + O(1) electronic steps + 1 optical step for prefix computation on the same number of processors and data points as considered in [4].**

***Index Terms*—Prefix computation, parallel algorithm, time complexity**.

I. INTRODUCTION

Optical Transpose Interconnection System (OTIS) [1],[2] is basically a hybrid architecture which is benefits from optical and electronic connections. Optical connection is used to connect the processors when the distance between the processors exceeds the few millimetres (in other package) and electronic connections are used to connect the close processors (within the same physical package). Several models exploit the idea of optical and electronic connections. Given N data values $x_1, x_2, \ldots, x_N$ and the associative binary operation o, the prefix problem is to compute $P_i = x_1 o x_2 o x_3 \ldots o x_i$, 1d"id"N. Depending upon the type of problem, the associative binary operation o perform arithmetic or logical operation. Throughout this paper we assume that associative binary operation perform as binary addition operation.

II. RELATED WORK

Many researchers has developed parallel algorithms for computing the prefix on several topologies. An n point parallel algorithm on two dimensional mesh with 3"N - 1communication steps can be found in [8]. Egecioglu and Srinivas[15] presented 2"N + 1communication steps and log N + 1 arithmetic steps, optimal algorithm also on mesh architecture. A parallel algorithm has been reported in [14] on an extended Multimesh that requires $13N^{1/4}$ communication steps(electronic) and log N + 4 arithmetic steps. Wang and sahani[7] developed a parallel algorithm on N-point prefix computation in $(8N^{1/4}-1)$ electronic and 2 OTIS moves for N processors on SIMD and MIMD model. Wang and sahani also modified the parallel algorithm in same paper and completes prefix computation in $(7N^{1/4}-1)$ electronic moves and 2 OTIS moves. Jana and sinha[19] developed a improved algorithm with $(5.5N^{1/4}+3)$ electronic moves and 2 OTIS moves on the same model. Jana and mallick developed a parallel prefix algorithm on OTIS mesh of trees[4] which can compute the prefix of N numbers in 3.5logN + O(1) electronic steps and it takes 2 OTIS moves. In this paper we propose here a improved parallel prefix algorithm on a OTIS-Mesh of Trees. We shown that the algorithm requires 10log n + O (1) electronic moves and 1 optical move. The algorithm is shown to be an improvement over the parallel prefix algorithm on OTIS–MOT[4]. The rest of the paper is organized as follows. Section III describes the topology of OTIS-Mesh of trees. In section IV ,we describe the parallel algorithm for prefix in each group as described in MOT[4] in subsection A and proposed parallel algorithm in subsection B.

III. TOPOLOGY

We describe here the topology of OTIS-MOT in three sections. Subsection A describe the topology of Mesh of Trees, subsection B describe the OTIS network and subsection C describe the computational model on which we map the parallel prefix algorithm.

A. Topology of Mesh Of Trees(MOT)

In n×n MOT, n^2 processor organized as an n×n lattice. In an n×n lattice each processor denotes by p(i,j) where i and j respectively are row and column and 1d" i , jd" n. Then the interconnectivity among the processors is described as follows:

i. The processor in the i^{th} row is connected to form binary tree from processors P(i,1). ie. for j=1 to $\left\lfloor \frac{n}{2} \right\rfloor$ processor P(i,j) is directly connected to the processor P(i,2j) and P(i,2j+1) whether they exists. We call such binary trees as row trees.

ii. The processor in the j^{th} column is connected to form binary tree from processors P(1,j). for i=1 to $\left\lfloor \frac{n}{2} \right\rfloor$ processor P(i,j) is directly connected to the processor P(2i , j) and P(2i+1, j) whether they exists. We call such binary trees as column trees.

iii. All links are bi-directional.

The topology of MOT shown in figure 1.

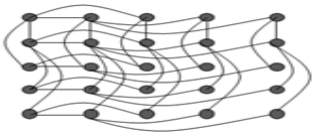

Figure 1. 5×5 Mesh Of Trees

DOI: 01.IJIT.01.03. 518

B. OTIS Network

Optical Transpose Interconnection System is basically a hybrid architecture which is benefits from optical and electronic connections. In this topology we use usual electronic links for communication between the processors within a group, and optical links for communication between the processors of other groups. According to the OTIS rule, G^{th} group is connected to the P^{th} processor and P^{th} group is connected to the G^{th} processor. The pattern of OTIS can be varied according to the interconnection of processors within a group, such as OTIS-MOT, OTIS-Hypercube, OTIS-Mesh etc. The topology of OTIS shown in figure 2.

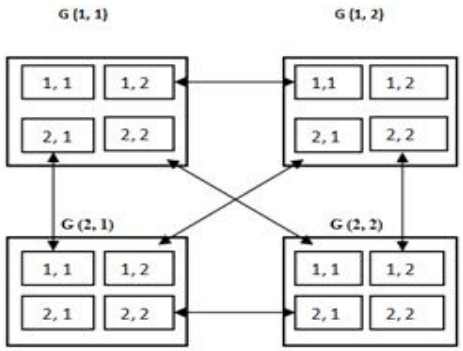

Figure 2. OTIS Network

C. Computational Model

In Optical Transpose Interconnection System, processors between the different groups are connected by optical links. n^4 number of processors are divided in to the n × n lattices where each lattice has n^2 processors. The processor placed at i^{th} row and j^{th} column within a lattice is denoted by P(i,j,k,l), where k and l denotes the lattice coordinates. Processors within a lattice are connected by the electronic links and processors placed at different lattices can be connected by optical links. According to the OTIS rule processor P(i,j,k,l) is connected to the P(k,l,i,j). Connectivity including electronic links and optical links shown in figure 3.

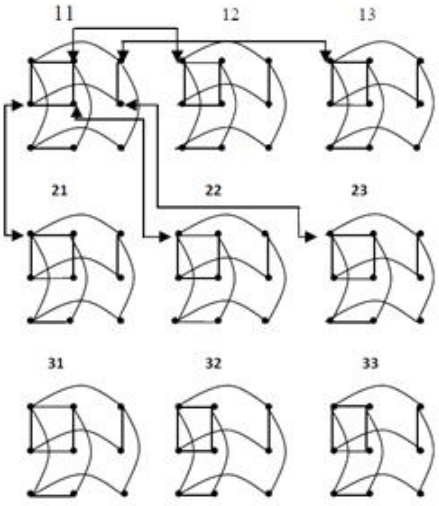

Figure 3. OTIS-Mesh Of Trees
All links are not shown and links are bidirectional

IV. PROPOSED PARALLEL PREFIX ALGORITHM

In proposed parallel prefix algorithm, first we describe parallel prefix algorithm in each group as considered in [4]. Then we describe the proposed parallel prefix algorithm on OTIS-Mesh of Trees on n^4 data points in section B.

A. Parallel Prefix in each group

Data initialization: n^2 data elements are stored in respective B(i,j) registers in row major order in each group. Register B(i,j) is initialized with the data element $x_{(i-1)n+(j-1)}$, 1d"i,jd"n for n =5 as we shown in figure 4.1.

X_0	X_1	X_2	X_3	X_4
X_5	X_6	X_7	X_8	X_9
X_{10}	X_{11}	X_{12}	X_{13}	X_{14}
X_{15}	X_{16}	X_{17}	X_{18}	X_{19}
X_{20}	X_{21}	X_{22}	X_{23}	X_{24}

Figure 4.1 Data initialization on lattice

step 1: Perform the prefix computation on each row in parallel and store the result in A(i,j) register as given in [8]. After this step register A(i,j) contains x[(i-1)n:(i-1)n+(j-1)], where x[p:q] shows the sum $x_p + x_{p+1} + ,...,+x_q$ for pd"q ,however we can see $x_p + x_{p+1} + ,...,+x_q$ is denoted by x_{p-q} in figure 4.2.

X_0	X_{0-1}	X_{0-2}	X_{0-3}	X_{0-4}
X_5	X_{5-6}	X_{5-7}	X_{5-8}	X_{5-9}
X_{10}	X_{10-11}	X_{10-12}	X_{10-1}	X_{10-14}
X_{15}	X_{15-16}	X_{15-17}	X_{15-18}	X_{15-19}
X_{20}	X_{20-21}	X_{20-22}	X_{20-23}	X_{20-24}

Figure 4.2 Data in A(i,j) register

***step 2*:**Perform prefix summation in each row tree on the contents of B(i,j) register in each row and store it in B(i,1) of the root processor P(i,1). B[i,1] holds x[(i-1)n + (j-1): in-1] after this step.

step 3: Perform modified prefix on the contents of B(i,1) of the 1st column tree using similar procedure as given in [8]. In this modified prefix, the processor P[i,1] computes the sum x[0:(i-1)n-1] for 2d"id"n and B[i,1] holds 0. The result is shown in figure 4.3 where "-" indicates the don't care value.

0	-	-	-	-
X_{0-4}	-	-	-	-
X_{0-9}	-	-	-	-
X_{0-14}	-	-	-	-
X_{0-19}	-	-	-	-

Figure 4.3 Modified Prefix on first column Tree

step 4: Broadcast the content of B(i,1) on each row as shown in figure 4.4.

DOI: 01.IJIT.01.03.518

Figure 4.4 Data in B(i,j) register after step 4

step 5: Add the contents of A(i,j) and B(i,j) registers for1d"i,jd"n and store the result in B(i,j) register for1d"i,jd"n to produce the final prefix. The content of B(i,j) register after step 5 shown in figure 4.5.

Figure 4.5 Data in B(i,j) register after step 5

Time complexity: The step1 and step3 requires log n steps each. Step2 and Step4 each require logn steps. Step5 requires no data movement. Therefore the above algorithm requires 4log n + O(1) steps.

B. Parallel Prefix on OTIS-MOT

*Data Initialization***.** First we store the data elements $x_0, x_1, x_2, x_3, \ldots, x_{n^4-1}$ in register B(i,j) following the row major order within the block and also in row major order from block to block. i.e B(i, j, k, l) is initialized with $x_{(i-1)n^3+(j-1)n^2+(k-1)n+(l-1)}$ for n = 3. Register C(i,j) is initialized with the value 0.

step 1: Compute the local prefix of each group stored in the register B(i,j) in parallel by applying the algorithm described in subsection A of section IV and store the result in register B(i,j,k,l) holds $x[(i-1)n^3+(j-1)n^2 : (i-1)n^3+(j-1)n^2+(k-1)n+(l-1)]$ which is shown in figure 4.6.

Figure 4.6 Data in B(i,j) register in each group

step 2:After computing local prefix, send the content of register B(n,n) to register A(n,n) of each group in parallel .

step 3: Broadcast the content from register A(n,n) to register A(i,j), 1d"i,jd"n through mesh of trees of each group in parallel. The content of register A(i,j) ,1d"i,jd"n for group 12 shown in figure 4.7.

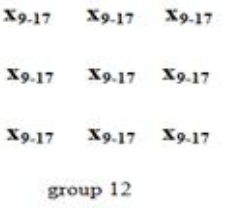

Figure 4.7 Content of register A (i,j) for group 12

*Rule 1***.** According to rule 1, processors in any group can send the data through valid OTIS links only to the processors in the successor groups. We assign group number to groups according to which we can decide which group is predecessor for any group and which group is successor for any group. Distributions of group numbers are in row major order 1 d" g d" n^2 where g is group number. Group with the large group number will be the successor of group with lower group number, for e.g. group 2 is successor of group 1. The distribution of group number shown in figure 4.8.

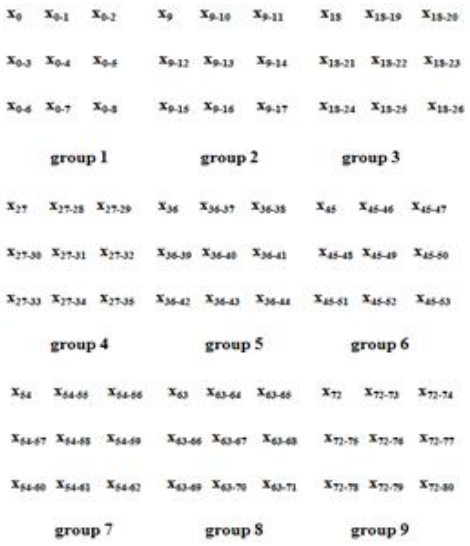

Figure 4.6 Data in B(i,j) register in each group

step 2:After computing local prefix, send the content of register B(n,n) to register A(n,n) of each group in parallel .

step 3: Broadcast the content from register A(n,n) to register A(i,j), 1d"i,jd"n through mesh of trees of each group in parallel. The content of register A(i,j) ,1d"i,jd"n for group 12 shown in figure 4.7.

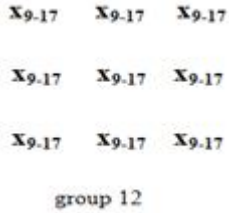

Figure 4.7 Content of register A (i,j) for group 12

DOI: 01.IJIT.01.03. 518

Rule 1. According to rule 1, processors in any group can send the data through valid OTIS links only to the processors in the successor groups. We assign group number to groups according to which we can decide which group is predecessor for any group and which group is successor for any group. Distributions of group numbers are in row major order $1 \leq g \leq n^2$ where g is group number. Group with the large group number will be the successor of group with lower group number, for e.g. group 2 is successor of group 1. The distribution of group number shown in figure 4.8.

Figure 4.8 Distribution of group numbers

step 4: After step 3 processors in each group sends the content from A(i,j) register, to C(i,j) register of processors of only successor groups, in parallel through OTIS links as considered in rule 1.The OTIS move from A(i,j) to C(i,j) register shown in figure 4.9.

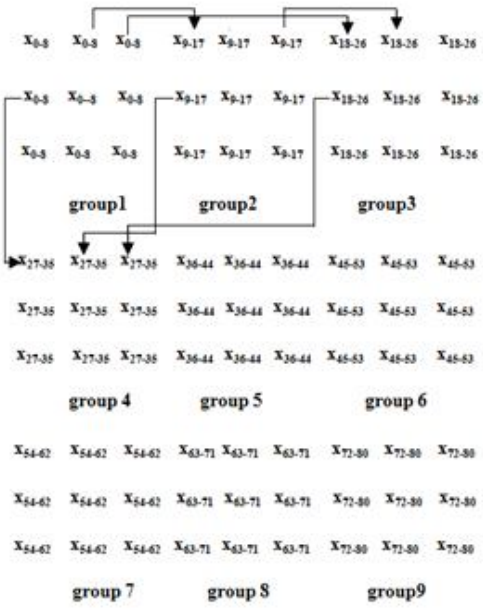

Figure 4.9 OTIS move from A(i,j) to C(i,j).
All moves are not shown and links are bidirectional

step 5: Perform prefix computation on each row on C(i,j) register of each group in parallel as given in [8] and store the result in C(i,j) register for $1 \leq i,j \leq n$.

step 6.After step 5 perform prefix computation on C(i,j) register of last column of each group in parallel as given in [8]. The content of C(i,j) registers of each group after step 6 is shown in figure 4.10.

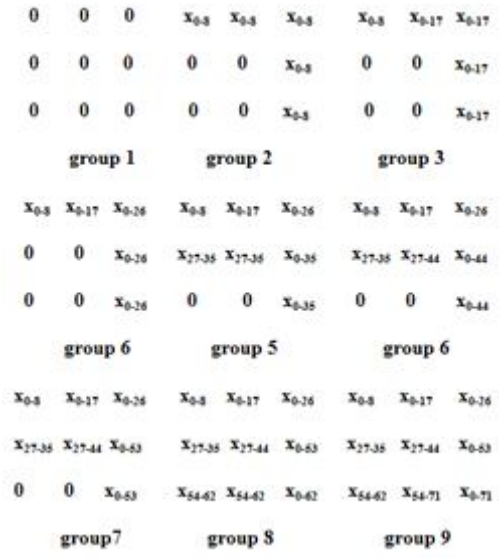

Figure 4.10 Content of C(i,j) register after step 6

step7: Broadcast the content from C(n,n) register to C(i,j) register for $1 \leq i,j \leq n$ of each group in parallel. The content of C(i,j) register of group 2 shown in fig 4.11 after step 7.

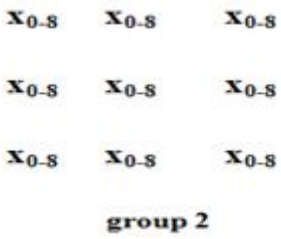

Figure 4.11 content of register C(i,j) for group 2

step 8: Finally add the content of C(i,j) and B(i,j) registers for $1 \leq i,j \leq n$ of each group in parallel and store the result in B(i,j) register for $1 \leq i,j \leq n$. The final result of prefix computation is shown in figure 4.12.

DOI: 01.IJIT.01.03. 518

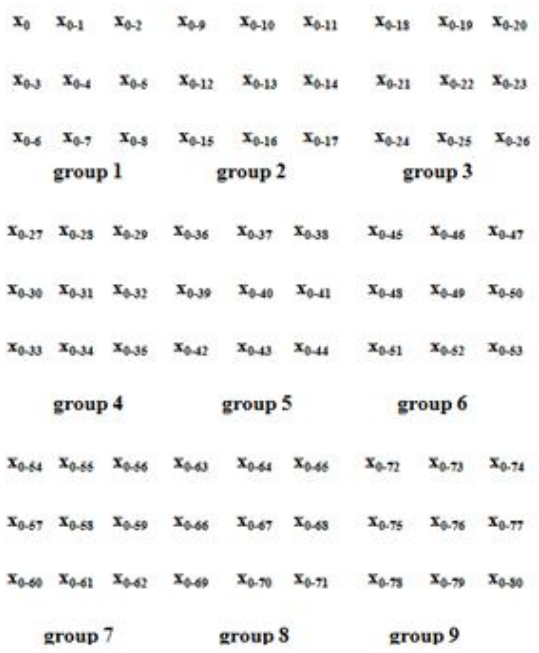

Figure 4.12 Data on B (i,j) register of each group

step 9: Stop

RESULTS

We describe here the time complexity required to map the parallel algorithm on OTIS-MOT on n^4 processors.

Time complexity: The algorithm described in subsection A of section IV requires 4log n + O(1) time for computing local prefix in each group in parallel as considered in MOT[4]. In our proposed algorithm, described in subsection B of section IV, Step2 takes O(1) time complexity. Step 3 takes 2log n time complexity for broadcasting. Step 4 requires single OTIS move .Step5 takes log n step for calculating prefix at each row of each group in parallel. Step6 takes logn time complexity. Step7 takes 2log n time complexity for broadcast. Step8 requires no data movement. Therefore the above algorithm requires 10log n + O(1) electronic moves + 1 OTIS optical move.

For N= n^4 above algorithm requires 2.5log N + O(1) electronic move + 1 OTIS optical move.

COMPARATIVE RESULT ANALYSIS

In Table I we compare the time complexity in terms of electronic moves and optical moves required by the parallel prefix algorithms mapped on OTIS network.

TABLE I.
COMPARISON OF OTIS-BASED PARALLEL ALGORITHM

Algorithm	Electronic Moves	OTIS Moves
Wang and sahani[7]	7n-1	2
Jana and Sinha[19]	5.5n+3	2
Dheeresh and Jana[4]	13log n + O(1)	2
Proposed Algorithm	10log n + O(1)	1

CONCLUSIONS

In this paper we have presented a new parallel prefix algorithm on OTIS-Mesh of Trees on a set of n^4 data points. The algorithm perform parallel prefix on OTIS-Mesh of trees in 10log n + O(1) electronic moves + 1 Optical move. This can be compared to earlier parallel prefix algorithm on OTIS-Mesh of Trees[4] on n^4 data points which requires 13log n + O(1) electronic moves + 2 optical move on same number of processors.

FUTURE WORKS

In future works we should try to map the parallel prefix algorithm on other interconnection networks so that we can perform parallel prefix computation in reduced time complexity. We can also propose a new interconnection network for mapping the parallel prefix algorithm.

REFERENCES

[1]. G.C Marsden, P.J Marchand, P. Harvey and S, C, Esener, "Optical Transpose Interconnection System Architecture," Optical Letters, Vol 18, No. 13,pp 1083-1085,july,1993.

[2]. C. F. Wang and S. Sahani, "OTIS optoelectronic computers Parallel Computation using Optical Interconnection," K Li, Y Pan and S.Q Zhang, Eds.Khuwer Academic 1988.

[3]. M. De, D. Das, B.P. Sinha, M. Ghosh, "An Efficient Sorting Algorithm on the Multi-Mesh," IEEE Trans. Comput. 46 (10) (1997).

[4]. Dheeresh K. Mallic, Prasanta K. Jana, "Parallel Prefix on Mesh of Trees and OTIS-Mesh Of Trees," int conf. Par, and Dis. Proc. Tech and appl |PDPTA 08|.

[5]. A.Osterloh."Sorting on the OTIS-Mesh," Proc 14th Int'l .Parallel and Distributed Processing Symposium(IPDPS 2000),pp 269-274, 2000.

[6]. P.K. Jana, B.P. Sinha, "Fast Parallel Algorithm for Polynomial Interpolation," Comput. Math. Applic. 29 (4) (1997) 85–92.

[7]. C.f. Wang and S. Sahani, "Basic operations on the OTIS-Mesh optoelectronic computer," IEEE Trans. On Parallel and Distributed System Vol9, No 12 pp 1226-1998, December 1998.[8] S.G. Akl, "Parallel Sorting Algorithms", Prentice-Hall, Orlando, FL, 1989.

[8]. S.G Akl. "The Design and Analysis of Parallel Algorithms," Englewood Cliffs, NJ Prentice Hall, 1989.

[9]. S. Sengupta, D. Das, B.P. Sinha, M. Ghosh "A Fast Parallel Algorithm for Polynomial Interpolation using Lagrange_s formula," in: Proceedings of the International Conference on High Performance Computing (HiPC), New Delhi, 1995, pp. 701–706.

[10]. P.K. Jana, B.P. Sinha, "Efficient Parallel Algorithms for Lagrange and Hermite interpolation," Int. J. Appl. Sci. Comput. 4 (2) (1998) 118–136.

[11]. D. Das, B.P. Sinha, "A New Network Topology with Multiple Meshes," Technical Report No. T/Rep/E-94/01, Indian Statistical Institute, Calcutta, 1999.

[12]. NAG, NAG Fortran Library Manual, Mark 13, N.A.G. Ltd., Oxford, 1988.

[13]. Harwell, Harwell Subroutine Library: a catalog of subroutines, Harwell Report AREA-R9185, Harwell,1989.

[14]. P. K. Jana, B. D. Naidu, S Kumar , M Arora and B.P. Sinha, "Parallel Prefix computation on Extended Multi-Mesh Network," Information Processing Letters, Vol 84, No, 6 .pp, 295-303,October 2002.

DOI: 01.IJIT.01.03. 518

[15]. O.Egecioglu and A Srinivasan ,"Optimal Parallel Prefix Mesh Architechture," Parallel Algorithm and Applications Vol. 1, 1993,pp 191-209.
[16]. P.K. Jana, "Finding Polynomial Zeros on a Multi-Mesh of Trees (MMT)," in: H. Mohanty, C. Baral (Eds.), Trends in Information Technology, Proceedings of the 2nd International Conference on Information Technology, (CIT _99), December 20–22, 1999, Bhubaneswar, India, Tata Mc- Graw-Hill, New Delhi, 2000, pp. 202–206.
[17]. P.K. Jana, "Fast Parallel Algorithms on Multi-Mesh of Trees (MMT)," in: B.P. Sinha, R. Gupta (Eds.), Recent Advances in Computing and Communications, Proceedings of the 8th International Conference on Advanced Computing and Communications, (ADCOM 2000), December 14–16, 2000, Cochin, India, Tata McGraw-Hill, New Delhi, 2000, pp. 221–227.
[18]. Q.F. Stout, "Mesh connected computers with Broadcasting," IEEE Trans. Comput. C-32 (1983) 826
[19]. P. K. Jana and B. P. Sinha "An Improved Parallel Prefix Algorithm on OTIS-Mesh," Parallel Processing Letters,pp-429-440,Vol.16,No. 4, 2006.

DOI: 01.IJIT.01.03.518